Lesson 6

The Control toolkit

Instructor:
ir drs E.J Boks
Elecrical Engineering
Embedded Systems Engineering
phone:(026) 3658173
Room: D2.03
e-mail:ewout.boks@han.nl

Control Toolbox overview

- The Control System Toolbox builds on the foundations of MATLAB to provide functions designed for control engineering.
- The Control System Toolbox is a collection of algorithms, written mostly as m-files, that implements common control system design, analysis, and modeling techniques.
- Convenient graphical user interfaces (GUIs) simplify typical control engineering tasks.

Control Toolbox overview

- Control systems can be modeled as transfer functions, in zero-pole-gain or state-space form, allowing you to use both classical and modern control techniques. You can manipulate both continuous-time and discrete-time systems. Conversions between various model representations are provided.
- Time responses, frequency responses, and root loci can be computed and graphed. Other functions allow pole placement, optimal control, and estimation. Finally, the Control System Toolbox is open and extensible. You can create custom M-files to suit your particular application.

Linear Models in the toolbox

- The Control System Toolbox supports the State Space model representation:
- State-space models (SS) of the form

$$\frac{dx}{dt} = Ax + Bu$$
$$y = Cx + du$$

- where A, B, C, and D are matrices of appropriate dimensions, x is the state vector, and u and y are the input and output vectors.

Model representations

- The Control System Toolbox supports the Transfer Function model representation:
- Transfer function (TF) example:

$$H(s) = \frac{s-4}{s^2+3s-7}$$

Model representations

- The Control System Toolbox supports the Zero-Pole-Gain representation:
- Zero-pole-gain (ZPK) model example :

$$H(z)=\frac{(z+2+3\mathrm{j})(z+2-3\mathrm{j})}{(z+3)(z-2)}$$

LTI Model representations

- The Control System Toolbox supports the Frequency Response Data representation (FRD).
- This model type is not further discussed here. Please consult your Control Toolbox manual for more information.

LTI object

- Internally, the Matlab Control Toolbox uses an general control object called the LTI object to represent the previously mentioned control model representations.
- LTI stands for *Linear Time Invariant* . A system is considered an LTI system when :
  - A signal y is the response to a signal x, then a*y is the response to a signal a*x.
  - A delay of the input x results in an equal delay of the output y.
- The LTI object is a transfer function (matrix) of the system represented.

Creating an LTI object

An LTI object of the type TF, ZPK, SS, or FRD is created whenever you invoke the corresponding constructor function, tf, zpk, ss, or frd. For example:

```
P = tf([1 2],[1 1 10]);
```

creates a TF object, P, that stores the numerator and denominator coefficients of the transfer function:

$$P(s)=\frac{s+2}{s^2+s+10}$$

This is a Single Input, Single Output (SISO) transfer function.

Creating a LTI object

You can also specify transfer functions as rational expressions in the Laplace variable s by

1. Defining the variable s as a special TF model :

```
s = tf('s');
```

2.Entering your transfer function as a rational expression in s.

For example, once s is defined with tf as in 1,

```
h = s/(s^2 + 2*s +10);
```

produces the same transfer function as

```
h = tf([1 0],[1 2 10]);
```

MIMO model creation

Multiple Input, Multiple Output (MIMO) models may be created most easily by forming a concatenation of SISO transfer function models.

Consider the following transfer function:

$$H(s)=\begin{bmatrix} \dfrac{s-1}{s+1} \\ \dfrac{s+2}{s^2+4s+5} \end{bmatrix}$$

```
h11 = tf([1 -1],[1 1]);

h21 = tf([1 2],[1 4 5]);

H = [h11 ; h21 ];
```

H is now the transfer function constructed out of h11 and h21.

Creating a pure gain model

- Pure gain models may be created by using tf with only one calling argument:

For example,

G = tf([1 0;2 1]);

produces the gain matrix:

$$G = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Creating Zero Pole Gain models

Similar to creating transfer function models using tf, zero pole gain models may be created using the zpf function.

For example, typing:

```
h = zpk(0, [1-j 1+j 2], -2);
```

creates the zero pole gain SISO model:

$$\frac{-2*s}{(s-2)(s-1+j)(s-1-j)}$$

MIMO models using zpf are created with the same procedure that was used with tf.

Creating State Space models

Use the command ss to create state-space models for : $\frac{dx}{dt} = Ax + Bu$

where u is input, x is the state and y is the output. $y = Cx + Du$

sys = ss(A,B,C,D);

For a model with Nx states, Ny outputs, and Nu inputs, the following applies:

- A is an Nx-by-Nx real-valued matrix.
- B is an Nx-by-Nu real-valued matrix.
- C is an Ny-by-Nx real-valued matrix.
- D is an Ny-by-Nu real-valued matrix.

This produces a State Space object sys that stores the state-space matrices A,B,C and D .

For models with a zero D matrix, you can use D = 0 (zero) as a shorthand for a zero matrix of the appropriate dimensions.

A State Space example

- Consider the following model of an electric motor:

$$\frac{d^2\theta}{dt^2} + 2\frac{d\theta}{dt} + 5\theta = 3\mathrm{I}$$

- θ is the angular displacement and I is the driving current of the motor. We would like to determine a state space model describing the relation between the input and the angular velocity of the motor.

State Space Example

- Since I is the input, we set u=I. The angular velocity is the output, so we set y=dθ/dt.
- Our state vector x describes the angular velocity and the angular displacement:

$$x = \begin{bmatrix} \theta \\ \frac{d\theta}{dt} \end{bmatrix}$$

- Then, our state vectors A, B, C and D are:

$$A = \begin{bmatrix} 0 & 1 \\ -5 & -2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \quad C = [0\ 3] \quad D = 0$$

State Space example

```
sys = ss([0 1;-5 -2],[0;3],[0 1],0);
```

to which MATLAB responds

```
a =
             x1        x2
   x1         0   1.00000
   x2  -5.00000  -2.00000

b =
             u1
   x1         0
   x2   3.00000

c =
             x1        x2
   y1         0   1.00000

d =
             u1
   y1         0
```

LTI object properties

- Each object has generic properties. Consult your Matlab manual for them.
- In addition, specific properties related to the model type exist.
- Each of these properties may be set with the set cmd, or retrieved with the get cmd.

Converting between models

- The Control Toolbox allows easy conversion between the different LTI model representations:
  - sys = tf(sys) Conversion to TF
  - sys = zpk(sys) Conversion to ZPK
  - sys = ss(sys) Conversion to SS
  - sys = frd(sys,frequency) Conversion to FRD
- FRD models cannot be converted to other models, but other models can be converted to FRD, provided a frequency is given.

Conversion example

A State Space example sys is created by:

```
sys = ss(-2,1,1,3);
```

to a zero-pole-gain model by typing

```
zpk(sys)
```

to which MATLAB responds:

```
Zero/pole/gain:
3 (s+2.333)
-----------
  (s+2)
```

Caution about conversions

- When manipulating or converting LTI models, keep in mind that:
- The three LTI model types TF, ZPK, and SS, are not equally well-suited for numerical computations. In particular, the accuracy of computations using high-order transfer functions is often poor. Therefore, it is often preferable to work with the state-space representation. In addition, it is often beneficial to balance and scale state-space models using ssbal. You get this type of balancing automatically when you convert any TF or ZPK model to state space using ss.
- Conversions to the transfer function representation using tf may incur a loss of accuracy. As a result, the transfer function poles may noticeably differ from the poles of the original zero-pole-gain or state-space model.

Caution about conversions

Conversions to state space are not uniquely defined in the SISO case, nor are they guaranteed to produce a minimal realization in the MIMO case. For a given state-space model sys,

```
ss(tf(sys))
```

may return a model with different state-space matrices, or even a different number of states in the MIMO case. Therefore, if possible, it is best to avoid converting back and forth between state-space and other model types.

Time delays

- Since LTI models are time invariant, delays may be introduced into any LTI model.
- Using the *ioDelay*, *InputDelay*, and *OutputDelay* properties of LTI objects, you can specify delays in both continuous- and discrete-time LTI models. With these properties, you can, for example, represent:
  - LTI models with independent delays for each input/output pair.
  - State-space models with delayed inputs and/or delayed outputs.

Operations on LTI models

You can perform basic matrix operations such as addition, multiplication, or concatenation on LTI models. Such operations are "overloaded," which means that they use the same syntax as they do for matrices, but are adapted so as to apply to the LTI model context.

The following operations are possible on LTI models:

- Precedence and Property Inheritance
- Extracting and Modifying Subsystems
- Arithmetic Operations
- Model Interconnection Functions
- Continuous/Discrete-Time Conversions of LTI Models
- Re-sampling of Discrete-Time Models

Model inheritance

- Operations like addition and commands like feedback operate on more than one LTI model at a time. If these LTI models are represented as LTI objects of different types (for example, the first operand is TF and the second operand is SS), it is not obvious what type (for example, TF or SS) the resulting model should be. Such type conflicts are resolved by precedence rules. Specifically, TF, ZPK, SS, and FRD objects are ranked according to the precedence hierarchy:

FRD > SS > ZPK > TF

An addition of two models sys1 (a TF model) and sys2 (a SS model) results in:

```
sys3 = sys1+sys2;
```

sys3 becomes a SS model as a result of the precedence rules.

Concatenation of LTI models

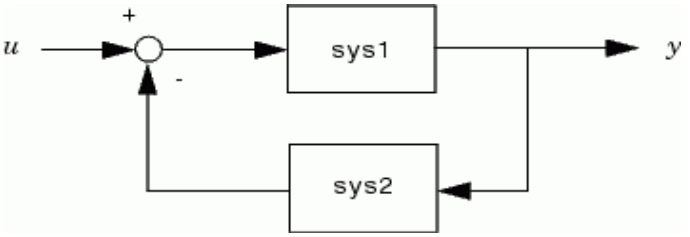
The Control System Toolbox provides a number of functions to help with the model building process. These include model interconnection functions to perform I/O concatenation ([,], [;], and append), general parallel and series connections (parallel and series), and feedback connections (feedback and lft). These functions are useful to model open- and closed-loop systems.

For example, when it is desired to create a feedback model of two systems sys1 and sys2, use the feedback cmd.

sys = feedback(sys1,sys2)

sys = feedback(sys1,sys2,sign)

sys = feedback(sys1,sys2,feedin,feedout,sign)



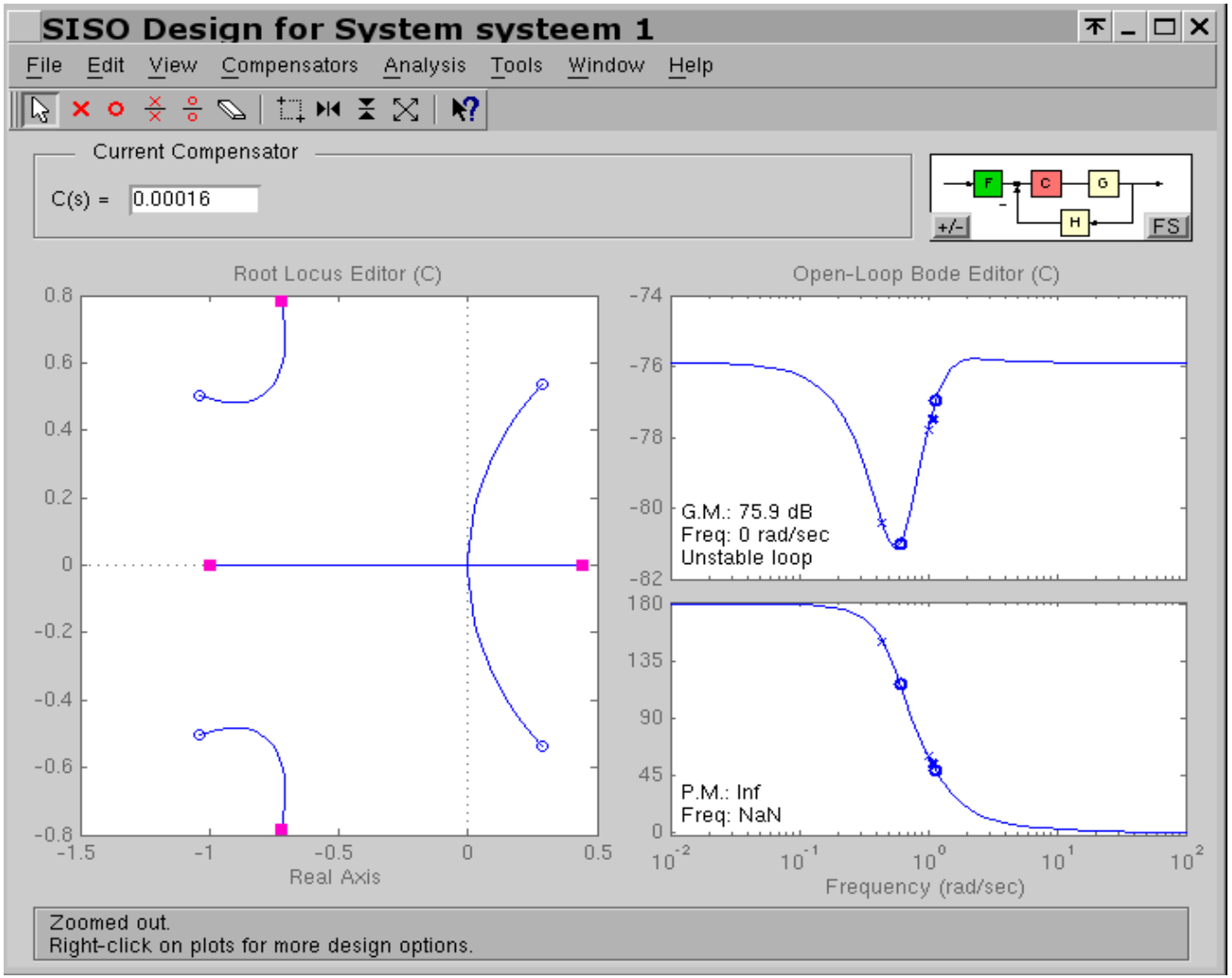
Conversion between continuous and discrete representation

The function c2d discretizes continuous-time TF, SS, or ZPK models. Conversely, d2c converts discrete-time TF, SS, or ZPK models to continuous time. Several discretization/interpolation methods are supported, including zero-order hold (ZOH), first-order hold (FOH), Tustin approximation with or without frequency prewarping, and matched poles and zeros.

LTI design tools

- To accelerate and facilitate the LTI design process, two GUI tools are available:
  - SISO design tool
  - LTI Viewer

The SISO design tool



The SISO design tool

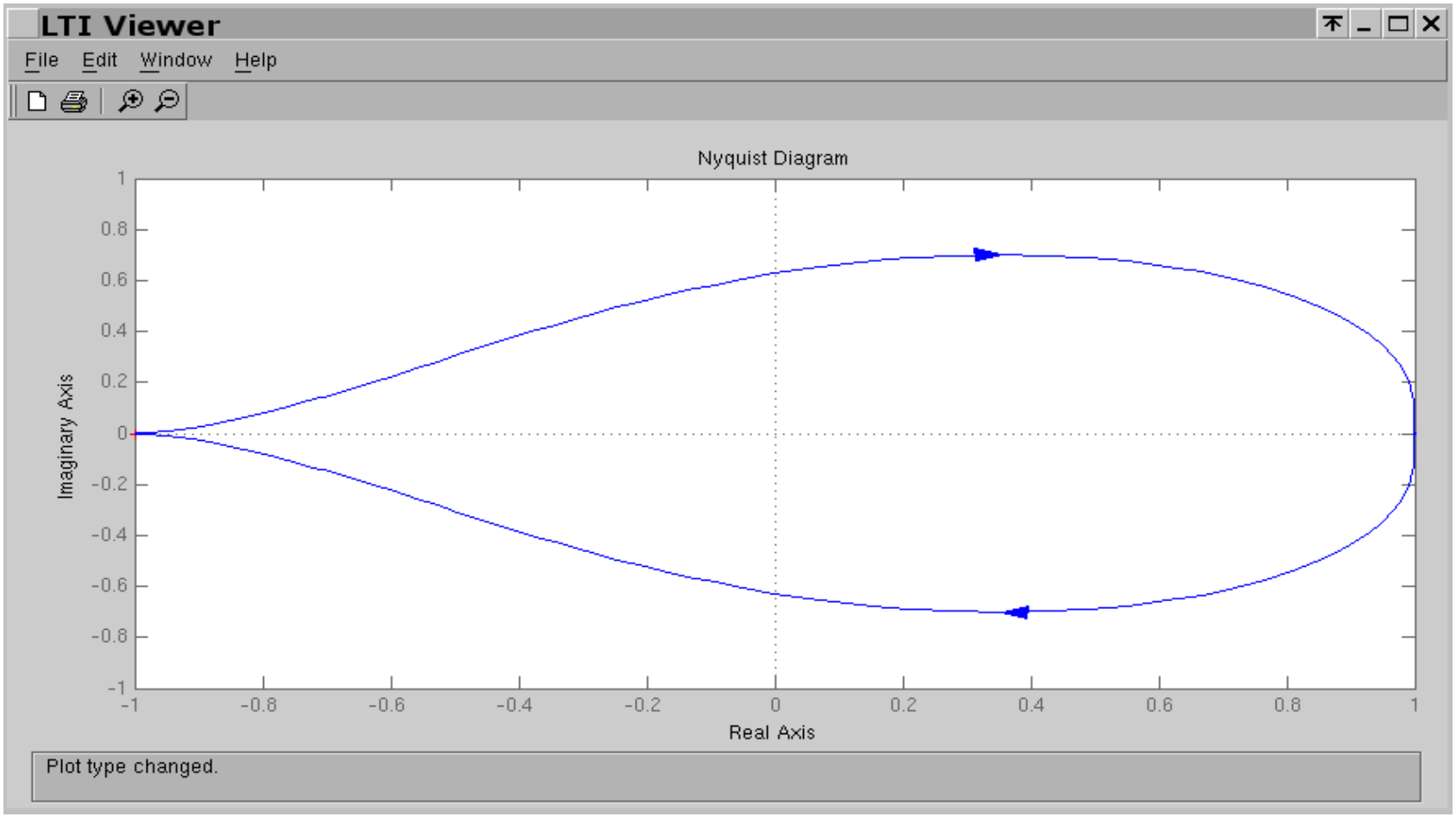
The SISO Design Tool is a graphical-user interface (GUI) that allows you to use root-locus, Bode diagram, and Nichols plot techniques to design compensators. The SISO Design Tool by default displays the root locus and Bode diagrams for your imported systems. The two are dynamically linked; for example, if you change the gain in the root locus, it immediately affects the Bode diagrams as well.

Opening the SISO Design Tool

Type: sisotool at the cmd prompt

to open the SISO Design Tool.

The LTI viewer



The LTI viewer

- The LTI Viewer is a graphical user interface (GUI) that supports ten plot responses, including step, impulse, Bode, Nyquist, Nichols, zero/pole, sigma (singular values), lsim, and initial plots. The latter two are only available at the initialization of the LTI Viewer; see ltiview for more information.
- The LTI Viewer is configurable and can display up to six plot type and any number of models in a single viewer. In addition, you can display information specific to the response plots, such as peak response, gain and phase margins, and so on.

The LTI Viewer can be started by typing: ltiview at the cmd prompt.

You can also open an LTI Viewer from the SISO Design Tool.

Control Toolbox exercise

- Consider an open loop transfer function :

$$H(s) = \frac{(s+1)}{s^3}$$

H(s) is inherently unstable. Design a closed loop system around H(s) using the SISO tool that is stable. Pick your system parameters (overshoot, phase margin etc) as you like.